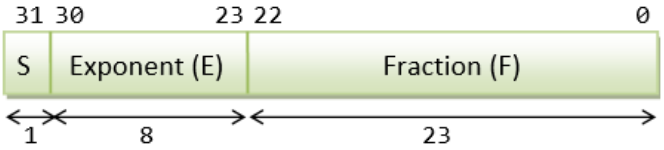
# IEEE-754 32-bit Single-Precision Floating-Point Numbers

In 32-bit single-precision floating-point representation:

- The most significant bit is the *sign bit* (S), with 0 for positive numbers and 1 for negative numbers.
- The following 8 bits represent *exponent* (E).
- The remaining 23 bits represents *fraction* (F).




**32-bit Single-Precision Floating-point Number**

## Part – I: Decimal to Floating Point Representation

Converting a number to floating point involves the following steps:

- **Set the sign bit** - if the number is positive, set the sign bit to 0. If the number is negative, set it to 1.
- **Divide your number into two sections** - the whole number part and the fraction part.
- **Convert to binary** - convert the two numbers into binary then join them together with a binary point.
- **Work out the exponent** - This is done by working out how many spaces the binary point needs to be moved so that it is just after the first **1** in the result. If you move the binary point to the left, then this number is positive. If you move it to the right, then the number is negative. Add 127 to this number then convert to binary. (Reason for adding 127: We need to represent both positive and negative exponent. With an 8-bit E, ranging from 0 to 255, the excess-127 scheme could provide actual exponent of -127 to 128)
- **Format the mantissa** - This is done by dropping the first **1** in the number and recording the next 23 bits.

Let us take examples and walk through these steps.

**Example 1** Find the 32-bit Floating Point representation of the number $(3.5)_{10}$

**Step 1**: Set the sign bit.
Since the number is positive, the sign bit S is 0.
**Step** 2: Divide your number in two sections.
3 is the whole number part and 5 is the fraction part.
**Step 3**: Convert to binary.
$(3)_{10} = (11)_2$
0.5*2 = 1.0 → Generate 1 and nothing remains.
So, $(3.5)_{10} = (11.1)_2$
**Step** 4: Work out the exponent.
$(11.1)_2 = (1.11 * 2^1)_2$
So, The exponent E is $(1+127)_{10} = (128)_{10} = (10000000)_2$
**Step** 5: Format the mantissa.
We will drop the 1 in 1.11 obtained in Step 4.

Hence, we have 11 and since we need 23 bits, we will extend zeroes in the right.
So, M = 11000000000000000000000
Hence, The Floating Point Representation is 0 10000000 11000000000000000000000

**Example 2** Convert -1313.3125 to IEEE 32-bit floating point format.

**Step 1**: Set the sign bit.
Since the number is negative, the sign bit S is 1.
**Step** 2: Divide your number in two sections.
1313 is the whole number part and 3125 is the fraction part.
**Step 3**: Convert to binary.
$(1313)_{10} = (10100100001)_2$

| | | | | |
|---|---|---|---|---|
| 0.3125 | × 2 = | 0.625 | 0 | Generate 0 and continue. |
| 0.62 | × 2 = | 1.25 | 1 | Generate 1 and continue with the rest. |
| 0.25 | × 2 = | 0.5 | 0 | Generate 0 and continue. |
| 0.5 | × 2 = | 1.0 | 1 | Generate 1 and nothing remains. |

Hence, $(1313.3125)_{10} = (10100100001.0101)_2$
**Step** 4: Work out the exponent.
$(10100100001.0101)_2 = (1.01001000010101 * 2^{10})_2$
So, The exponent E is $(10+127)_{10} = (137)_{10} = (10001001)_2$
**Step** 5: Format the mantissa.
We will drop the 1 in 1.01001000010101 obtained in Step 4.
Hence, we have 01001000010101 and since we need 23 bits, we will extend zeroes in the right.
So, M = 01001000010101000000000
Hence, The Floating Point Representation is 1 10001001 01001000010101000000000

## Part – II: Floating Point Representation to Decimal

Converting a number in 32-bit floating point representation to decimal involves the following steps:

- **Separate into the sign, exponent and mantissa fields** – Take the first bit as sign bit S, the next 8 bits as exponent E and the remaining 23 bits as the mantissa M.
- **Mantissa** – Extract the mantissa from the mantissa field and restore the leading one (Remember? We dropped the first 1 in Step 5 of our conversion from Decimal to FP). You may also omit the trailing zeroes.
- **Exponent** – Extract the exponent from the exponent field. Now, subtract the bias (127 in the 32-bit FP representation) to recover the actual exponent of 2 (Remember? We added 127 while working out the exponent in Step 4 of our conversion from FP to Decimal).
- **De-normalize the number** – Move the binary point so the exponent is 0, and the value of the number remains unchanged.
- **Convert the binary value to decimal** – This is done using the normal conversion process.
- **Set the sign of the decimal number** – If the sign bit S obtained in step 1 is 0, make the number positive, otherwise negative.

Let us take examples and walk through these steps.

**Example 3** Convert the 32-bit FP number 0 10001000 01101100001000000000000 to Decimal.
**Step 1**: Separate into the sign, exponent and mantissa fields
S = 0
E = 10001000
M = 01101100001000000000000
**Step 2**: Mantissa
Restoring the leading one in Mantissa, we get 1.01101100001000000000000
Now, omitting the trailing zeroes, we get 1.01101100001
**Step 3**: Exponent
$(10001000)_2 = (136)_{10}$
After subtracting the bias 127, we get 9 (136-127 = 9).
So, the actual exponent is 9.
**Step 4**: De-normalize the number
$(1.01101100001 * 2^9)_2 = (1011011000.01)_2$
**Step 5**: Convert the binary value to decimal
$(1011011000.01)_2 = (728.25)_{10}$
**Step 6**: Set the sign of the decimal number
Since S=0, sign will be positive.
Hence, the Decimal number is 728.25

**Example 4** Convert the 32-bit FP number 1 01000110 01101011000000000000000 to Decimal.
**Step 1**: Separate into the sign, exponent and mantissa fields
S = 1
E = 01000110
M = 01101011000000000000000
**Step 2**: Mantissa
Restoring the leading one in Mantissa, we get 1.01101011000000000000000
Now, omitting the trailing zeroes, we get 1.01101011
**Step 3**: Exponent
$(01000110)_2 = (70)_{10}$
After subtracting the bias 127, we get -57 (70-127 = -57).
So, the actual exponent is -57
**Step 4 & 5**: De-normalizing the number and conversion of binary to decimal
Since the exponent is far from zero, we will convert the original (normalized) mantissa.
$(1.01101011)_2 = (1.41796875)_{10}$
$(1.01101011 * 2^{-57})_2 = (1.41796875 * 2^{-57})_{10}$
$(1.41796875 * 2^{-57})_{10} = (9.83913471531 * 10^{-18})_{10}$ → Simple calculation using calculator
**Step 6**: Set the sign of the decimal number
Since S=1, sign will be negative.
Hence, the Decimal number is $-9.83913471531 * 10^{-18}$

**References:**
http://sandbox.mc.edu/~bennet/cs110/flt/